What is Random Forests?

RF is an classification and regression ensemble learning method that constructs a multitude of decision trees at training time and outputs the class that is the mode of the classes output by individual trees. It has few parameters to tune and can be used efficiently with default settings (i.e. they are effectively non-parametric). RF are good to use as a first cut when you don't know the underlying model, or when you need to produce a decent model under time pressure.

This ease of use allows people lacking a background in statistics to produce fairly strong predictions free from many common mistakes, with only a small amount of research and programming.

CRAN Task View: Machine Learning & Statistical Learning (7/31/13) http://cran.r-project.org/web/views/MachineLearning.html
Some Practical Reasons for using it.

Preprocessing is not needed:
Decision Trees do not care if the data is continuous, discrete, or contains character values. You won’t have to scale variables to have a mean of 0 and a standard deviation of 1.

Classification and regression are easy:
The same dataset can be used in times when it’s not clear if you want to use regression or classification (boolean output problem).

Easy tuning:
Parameters: mtry & ntree. Address overfitting with extremely randomized trees. Variable importances is easy.

Great Parallelization
null
classification

SVC
Ensemble Classifiers

kernel approximation

NOT WORKING

NOT WORKING

KNeighbors Classifier

SGD Classifier

<100K samples

Naive Bayes

Text Data

Linear SVC

<100K samples

START

get more data

>50 samples

regression

SGD Regressor

predicting a category

do you have labeled data

few features should be important

<100K samples

predicting a quantity

just looking

Randomized PCA

number of categories known

<10K samples

<10K samples

MiniBatch KMeans

Just looking

<10K samples

<10K samples

MeanShift

NOT WORKING

NOT WORKING

NOT WORKING

Spectral Clustering

GMM

KMeans

MiniBatch KMeans

NOT WORKING
Matplotlib


```
In [1]: from pylab import *
X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C, S = np.cos(X), np.sin(X)
plot(X, C)
plot(X, S)
show()
```
Decision Trees

Can be used for classification or regression

Node splits on any of the attributes

Prune tree to avoid over-fitting

Given **one** data set, get **one** predictor
The Strong Law of Large Numbers

The sample average converges almost surely to the expected value

$$\bar{X}_n \xrightarrow{a.s.} \mu \quad \text{when } n \to \infty.$$ 

This result explains why random forests do not overfit as more trees are added, but produce a limiting value of the generalization error.

http://www.stat.berkeley.edu/~breiman/random-forests.pdf
Bootstrapping

Tool for inference when the underlying distribution is unknown

Create replicate data sets from one data set

Sample N times with replacement from a data set of N observations → one replicate

Similar to drawing samples from the same underlying distribution

```r
sample(x, size, replace, prob)
library(boot)
boot(data= , statistic= , R=, ...)
```
Bootstrapping

Resample the sample with replacement
Bagging

**Bagging** = **Bootstrap Aggregating**

Use bootstrapping to generate a lot of replicate data sets.

Grow a decision tree for each replicate.

Aggregate tree predictions to get a forest prediction.

library(ipred)
Bagging

Data

Replicates

New Data → $T_1 + T_2 + T_3 + ...$ → Prediction
Ensemble / Aggregate

Average

Weighted average

Vote

Weighted vote

Others...
Bagging Mechanics

**Unstable:**
ANNs
classification & regression trees
Subset selection (linear regression)

**Stable:**
kNN
Random Forests

Bagging of trees
+ Random variable selection

= Random Forest

**Input:** one data set

**Bootstrap:** many replicates

**Train:** a decision tree for each replicate
*At each node, use one of “mtry” randomly sampled variables*

**Predict:** new data through aggregate

library(randomForest)
library(party)
library(randomForestSRC)
Tunable Parameters

**mtry**: randomly selected sample variable for each split

**ntree**: number of trees (default: 500)

**nodesize**: minimum size leaf nodes

**sampsize**: number of observation for each replicate (N)

**importance** - variable importance (default: False)

library(randomForest)
“Out of bag”

For each replicate, data not sampled is "out of bag" (OOB)

Can estimate error rate from OOB data

Proximity measure

Build each tree to maximum depth
ML resources

CRAN Task View: Machine Learning & Statistical Learning (7/31/13)
http://cran.r-project.org/web/views/MachineLearning.html

A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data
http://www.biomedcentral.com/1471-2105/10/213